WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Thursday, September 16, 2004

Hide?	Set Name	Query	Hit Count
		DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ	
☐	L11	L10 and ((function or functional) near3 call)	2
☐	L10	L9 and X-window	4
☐	L9	Xt adj3 library	18
☐	L8	L7 and (supplant or supplanting or supplanted)	2
☐	L7	Motif adj4 dialog	14
☐	L6	Motif	33449
☐	L5	L3 and (supplant or supplanting or supplanted)	2
☐	L4	L3 and (supplant or supplanting or supplanted)	2
☐	L3	Motif adj2 dialog adj2 box	5
☐	L2	Motif adj2 bialog adj2 box	0
		DB=USPT; PLUR=YES; OP=ADJ	
☐	L1	6118446.pn.	1

END OF SEARCH HISTORY

L10: Entry 2 of 4 File: USPT Apr 28, 1998

DOCUMENT-IDENTIFIER: US 5745115 A
TITLE: Graphical user interface having a shared menu bar for opened applications

Brief Summary Text (8):
OpenDoc is being implemented across various platforms of operating systems, such as, for example, the Unix-based operating system developed by IBM Corporation known as AIX, which uses the X-Windows graphics system. Within the cross-platform Unix-based system is a GUI toolkit known as Motif. The Motif toolkit does not provide a menu manager or shared menu bar. The Motif applications create one Motif menu bar at the top of the main application window. This menu bar is created with all the menu items needed by that application and it is not altered during the execution of the application. The changing menu bar is not necessary in this environment since only one application is using the menu bar.

Brief Summary Text (16):
Further to the preferred embodiment, an OpenDoc system is operated in the X Window environment and includes an OpenDoc menu library that communicates with the part editor for each active application. The menu library further communicates with the Motif widget library, which in turn communicates with an Xt toolkit library. The Xt toolkit library communicates with an Xlib graphics library that provides the screen drawing calculations for the display in the computer system.

Detailed Description Text (13):
An example of such is shown in FIG. 3. In FIG. 3 is shown a GUI 202, where application A and application B are shown. Application A 204 includes a menu bar 206 while application B 208 includes a menu bar 210. Motif API calls and Motif callbacks are exchanged between application A and the Motif widget library 212 which further exchanges API calls and callbacks between XT toolkit library 214. The toolkit library 214 exchanges X API calls and XEvents with the Xlib graphics library 216, which provides screen display in the GUI 202. The same operation occurs for application B 208, which operates independently of the application A 204.

Detailed Description Text (17):
The OpenDoc menu library 302 then receives Motif callbacks from and sends Motif API calls to Motif widget library 212. Motif widget library 212 receives Xt callbacks from and sends Xt API calls to Xt toolkit library 214. Xt toolkit library 214 also receives XEvents from and sends X API calls to the Xlib graphics library 216. The graphics library 216 sends and receives these calls and events from and to GUI 200. Shown in GUI 200 is a single window 310 where software component A 312 and software component B 314, represented by part A and part B, respectively, are illustrated that both share a common OpenDoc shared or base menu bar 316.

CLAIMS:

7. The method of claim 6 wherein said Motif widget library performs the steps of:

sending Xt API calls to an Xt toolkit library; and receiving Xt callbacks from said Xt toolkit library.

8. The method of claim 7 wherein said Xt toolkit library performs the steps of:

sending X API calls to an Xlib graphics library; and receiving X events from said Xlib graphics library.

20. The computer system of claim 19 wherein said Motif widget library performs the steps of:

sending Xt API calls to an Xt toolkit library; and

receiving Xt callbacks from said Xt toolkit library.

21. The computer system of claim 20 wherein said Xt toolkit library performs the steps of:

sending X API calls to an Xlib graphics library; and

receiving X events from said Xlib graphics library.

Previous Doc Next Doc Go to Doc#

L10: Entry 3 of 4 File: USPT Jan 21, 1997

DOCUMENT-IDENTIFIER: US 5596714 A
TITLE: Method for simultaneously testing multiple graphic user interface programs

Brief Summary Text (4):
In the field of computer technology, as microprocessors employing advanced processing architectures capable of greater computing power have been developed, both operating system software and application software have also been developed which take advantage of and, in many cases require, such increased power. Much of this advanced software development has been for the purpose of increasing the `user friendliness` of computers through the deployment of software having a more intuitive Graphic User Interface (GUI). The use of GUI's has been steadily increasing on almost every processing platform including Windows (in each of its various flavors) for the IBM-PC and System 7 for the Apple Macintosh computer. In addition, there has also been developed a class of GUI's which are not computer specific, but are rather network based display architectures. One such example of a current popular network based display architecture is the X-Window system developed at MIT and currently in release 11, (heretofore known as X11). X11 is particularly well regarded for its network transparency; network independence; device independence; and flexible user interface. In addition X11 is non-proprietary, and while it is most frequently run on UNIX computers, is also widely available to run on a wide range of other computer systems.

Brief Summary Text (11):
Recently, there has been developed an architecture which permits multiple applications to be tested at the same time. However, in order to accomplish this a special X11 server is required, which means that the computer code which is shipped after successful testing is not exactly the same as the computer code which has been tested. In addition, these multi-application testers require relinking the code of the tested program with the standard Xt library, which also increases the difference between the code tested and the code shipped.

Brief Summary Text (18):
3. The exact binary code which has been tested is the same code which may be shipped to customers, and no relinking with the standard Xt library is needed;

Detailed Description Text (28):
In addition, before performing a comparison test it is important to synchronize the output of the AUT with the X11 server and the terminal in order to make sure that the system is in a known state prior to Test. Even though the AUT can technically synchronize itself with the X11 server and the terminal, there is no mechanism in the X11 server (or in the X-Windows standard) for requesting the AUT to do this. This means that with the configuration shown in FIG. 2, the GUI Tester has no method available to support synchronization other than by waiting a predetermined amount of time. Since X servers are traditionally supplied by terminal vendors, modifying the X server itself is not a feasible solution. It is also not a desirable solution since, under ideal conditions the GUI Test architecture will match the end-user's architecture exactly in order to provide the highest level of confidence that a GUI Test will simulate real world operating conditions. When the X server is modified to accommodate test procedures it no longer represents with

the same exactness the X servers found in the field.

Previous Doc Next Doc Go to Doc#